Projekt

Sterowniki robotów

Dokumentacja

Humanistycznie upośledzony robot akrobatyczny

HURA

Skład grupy:
Albert Lis, 235534
Michał Moruń, 235986

Termin: sr TP15

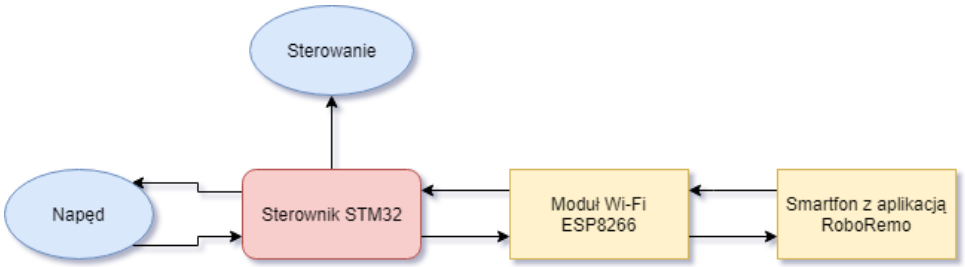
Prowadzący:
mgr inż. Wojciech Domski

2 czerwca 2019

Spis treści

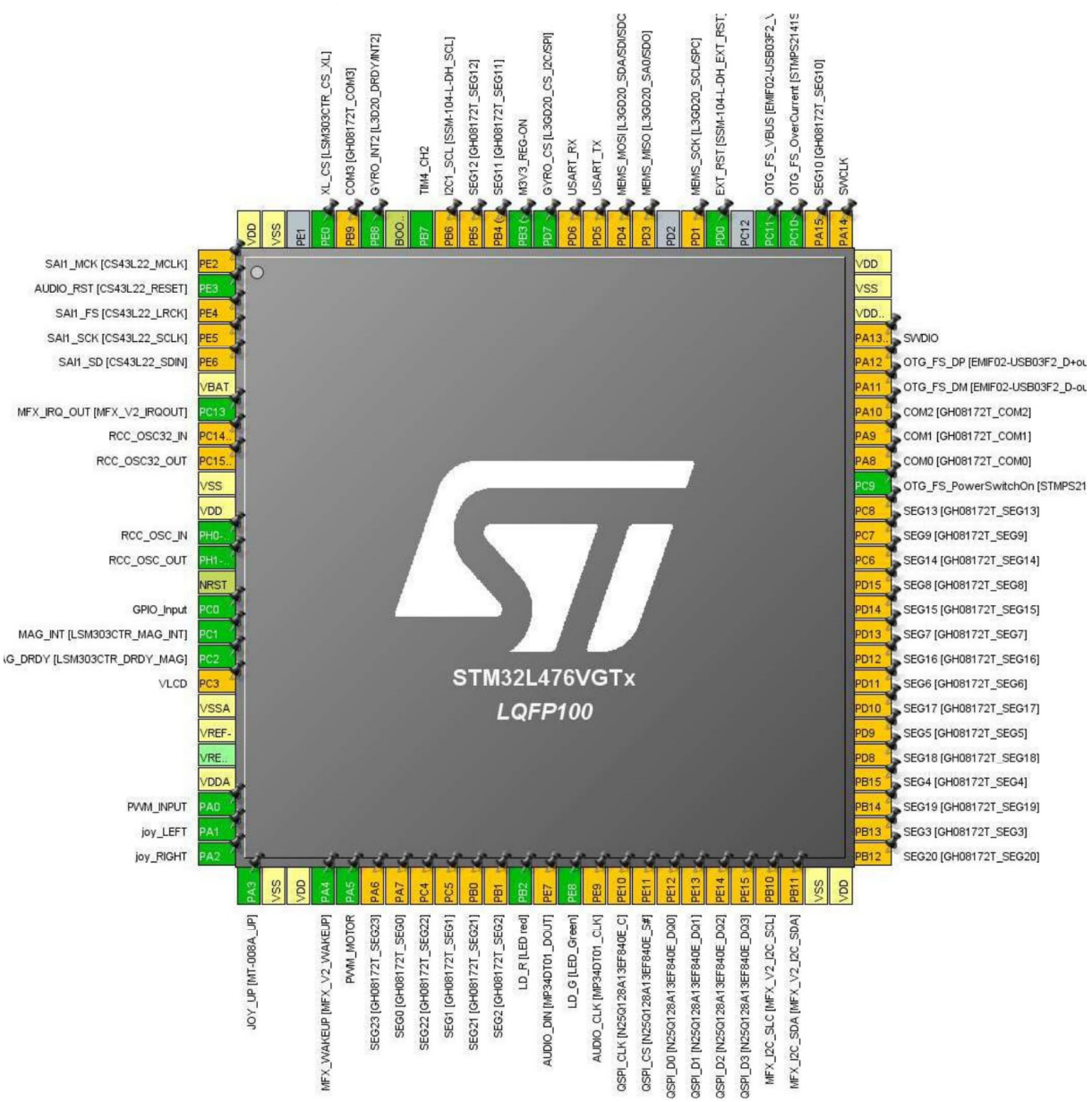
1 Opis projektu

Celem projektu jest zbudowanie zdalnie sterowanego robota jezdnego. Robot będzie sterowany za pomocą akcelerometru w telefonie. Dane będą przesyłanie za pomocą Wi-Fi lub Bluetooth. Regulacja prędkości będzie się odbywać za pomocą regulatora PID. Dane o prędkości będą pobierane z enkoderów znajdujących się w kołach robota. Opcjonalnie robot będzie wyświetlał szczegółowe dane o swoim stanie wewnętrznym za pomocą wbudowanego w płytkę z mikrokontrolerem wyświetlacza LCD.

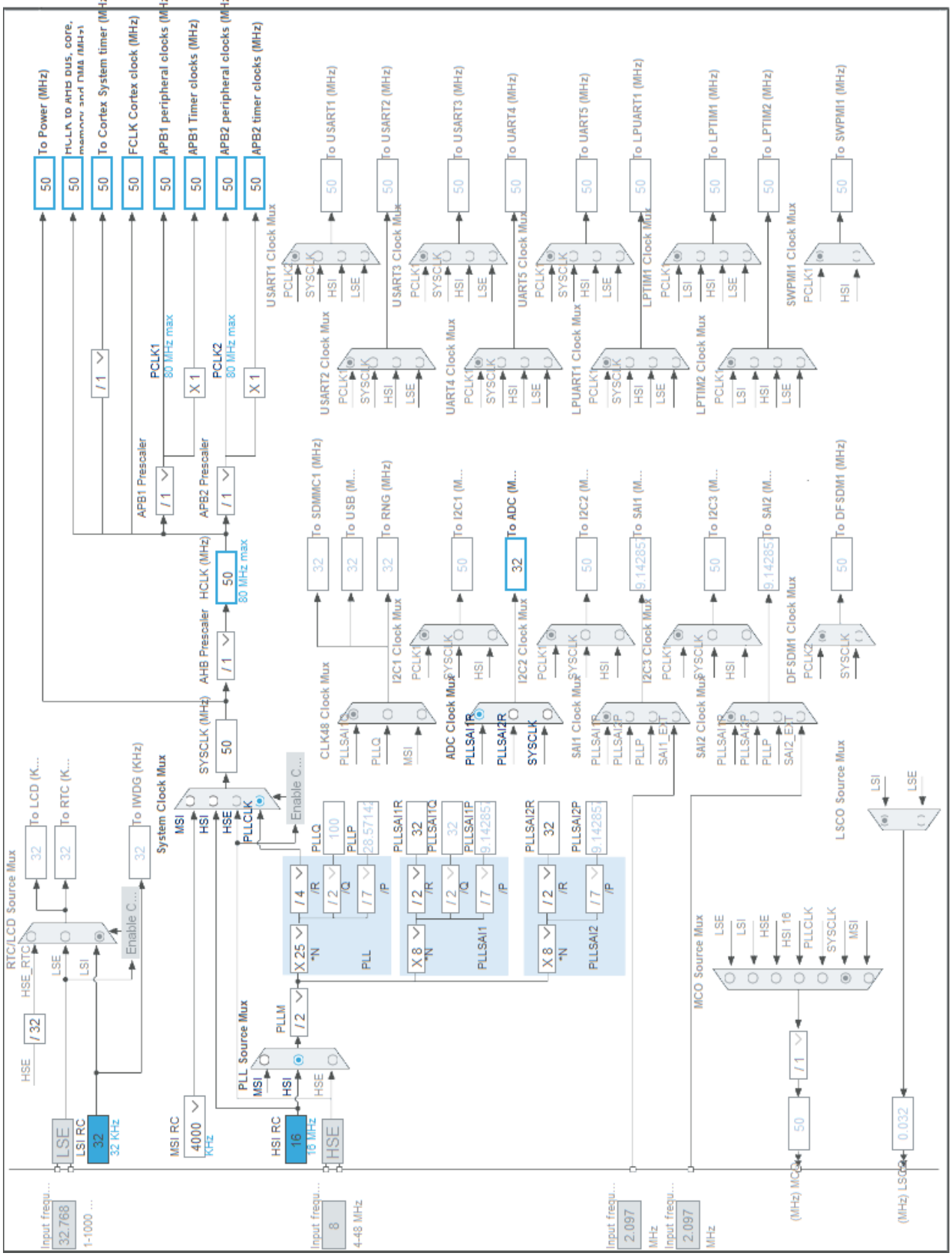


Rysunek 1: Architektura systemu

2 Konfiguracja mikrokontrolera



Rysunek 2: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 3: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

PIN	Tryb pracy	Funkcja/etykieta
PC14	OSC32_IN* RCC_OSC32_IN	
PC15	OSC32_OUT* RCC_OSC32_OUT	
PH0	OSC_IN* RCC_OSC_IN	
PH1	OSC_OUT*	RCC_OSC_OUT
PA2	USART2_TX	USART_TX
PA3	USART2_RX	USART_RX
PA5	TIM2_CH1	PWM_MOTOR
PB7	TIM4_CH2	PWM_SERVO

Tabela 1: Konfiguracja pinów mikrokontrolera

2.2 USART

Interfejs jest wykorzystywany do komunikacji z modułem Wi-Fi (ESP8266). Moduł odbiera dane za pomocą interfejsu UDP i przekazuje je do mikrokontrolera STM32 za pomocą interfejsu komunikacji szeregowej.

Parametr	Wartość
Baud Rate	115200
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1

Tabela 2: Konfiguracja peryferium USART

2.3 Timer 2

Parametr	Wartość
Clock Source	Internal Clock
Channel1	PWM Generation CH1
Prescaler	PWM_PRESC
Counter Mode	Up
Counter Period	PWM_PERIOD
Internal Clock Division	No Division
Mode	PWM mode 1
CH Polarity	High

Tabela 3: Konfiguracja peryferium Timer 2

2.4 Timer 2

Parametr	Wartość
Clock Source	Internal Clock
Channel	PWM Generation CH2
Prescaler	999
Counter Mode	Up
Counter Period	999
Internal Clock Division	No Division
Mode	PWM mode 1
CH Polarity	High

Tabela 4: Konfiguracja peryferium Timer 4

2.5 Timer 5

Parametr	Wartość
Prescaler	TIM5_PRESC
Counter Mode	Up
Counter Period	TIM5_PERIOD
Trigger Event Selection	Update Event

Tabela 5: Konfiguracja peryferium Timer 6

3 Urządzenia zewnętrzne

Rozdział ten powinien zawierać opis i konfigurację zewnętrznych, jak np. akcelerometr.

3.1 Moduł Wi-Fi ESP8266

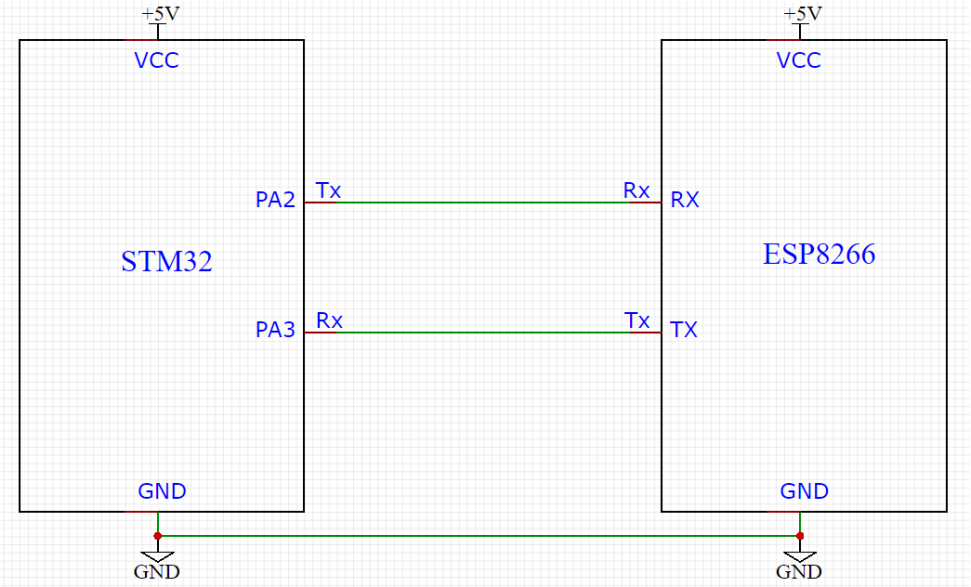
Moduł dobiera dane za pomocą protokołu UDP i przekazuje je odpowiednio sformatowane za pomocą portu szeregowego do mikrokontrolera STM32. Do oprogramowania modułu wykorzystano framework Arduino.
Ustawienia komunikacji szeregowej:

Parametr	Wartość
Baud Rate	115200
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1

Tabela 6: Konfiguracja peryferium UART w module Wi-Fi

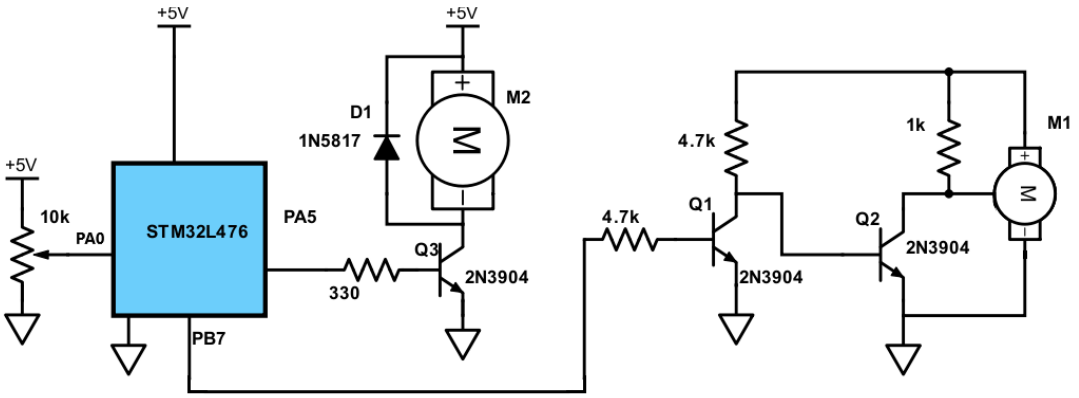
4 Projekt elektroniki

4.1 Połączenie pomiędzy STM32 a modułem WiFi



Rysunek 4: Połączenie STM32 z ESP8266

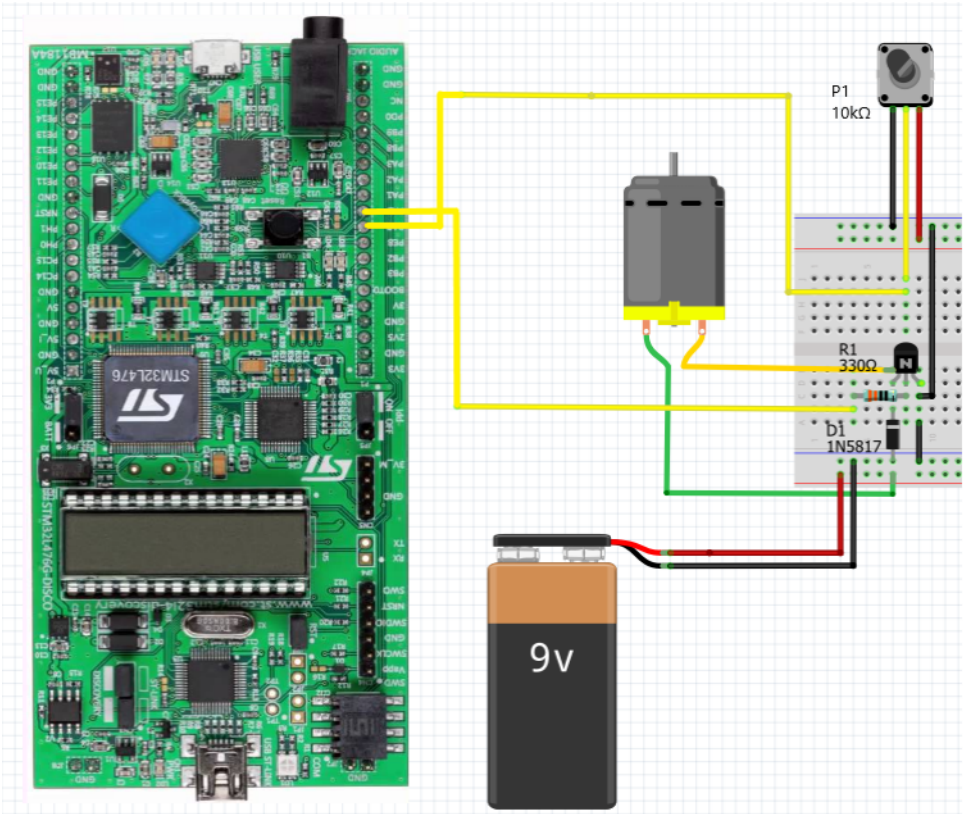
4.2 Schemat połączenia z serwomechanizmem



Rysunek 5: Schemat elektryczny

4.3 Schemat połączenia z silnikiem

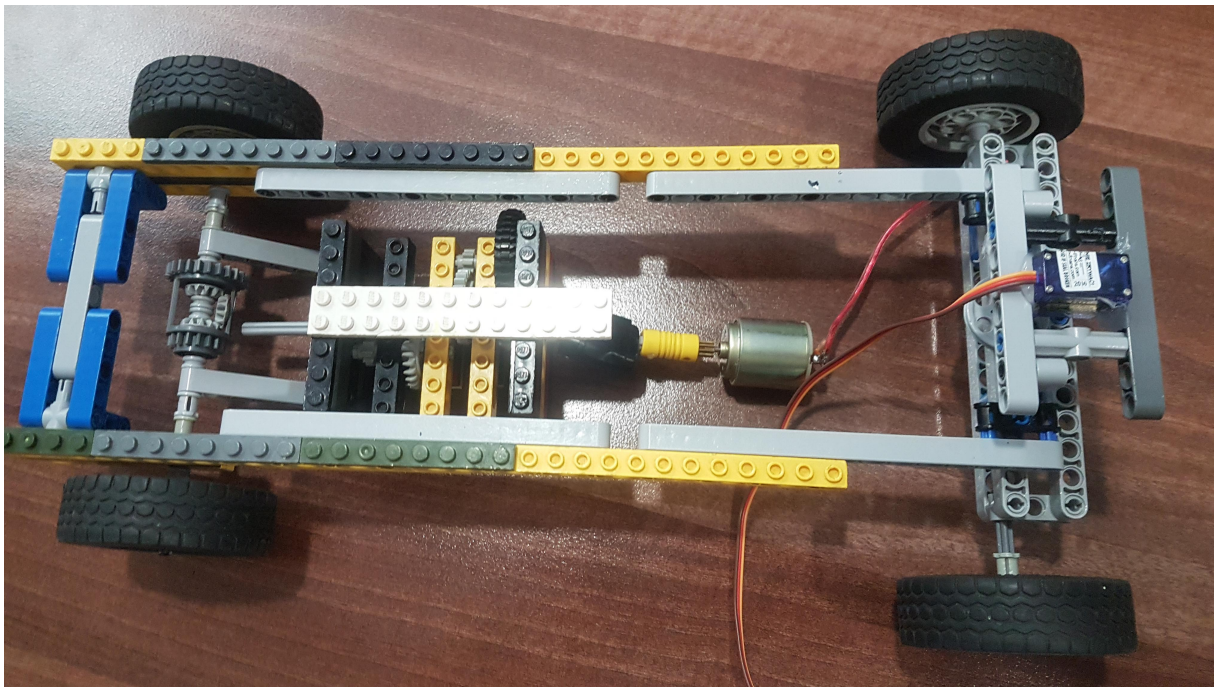
Za pomocą potencjometru regulujemy wypełnienie sygnału PWM. Sygnał ten jest wzmacniany za pomocą tranzystora NPN i przekazywany do silnika DC.



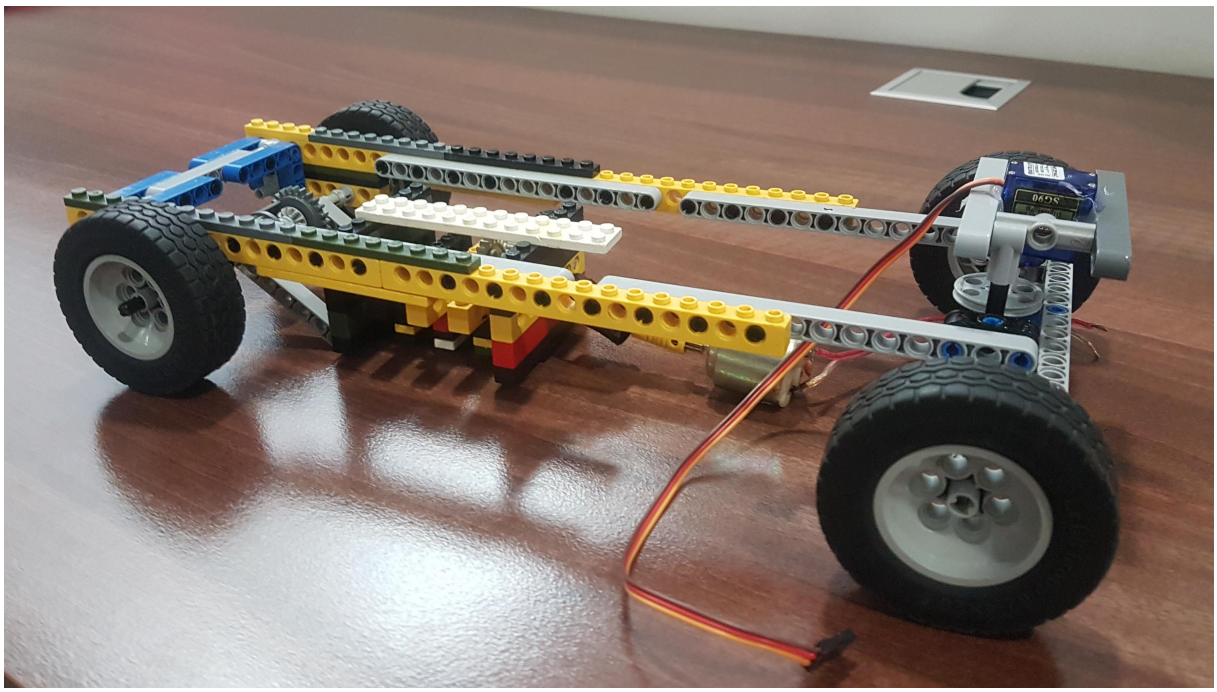
Rysunek 6: Schemat poglądowy regulacji prędkości obrotowej silnika

4.4 Schemat połączenia z enkoderem

5 Konstrukcja mechaniczna



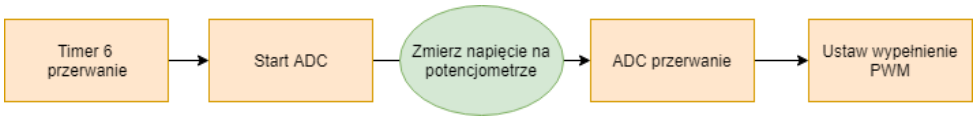
Rysunek 7: Zdjęcie części mechanicznej nr 1



Rysunek 8: Zdjęcie części mechanicznej nr 2

6 Opis działania programu

6.1 Schemat działania programu



Rysunek 9: Schemat działania programu

6.2 Funkcja obsługująca przerwanie timera 6

```
1  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
2  {
3    if(htim->Instance == TIM6)
4      HAL_ADC_Start_DMA(&hadc1, (uint32_t *)&adc_value, 1);
5  }
```

6.3 Funkcja obsługująca przerwanie USART

```
1  volatile uint8_t xvalue, yvalue, yprevious, xprevious;
2  volatile char xsign;
3  void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
4  {
5    axis = (char)(Received[nrAxis]);
6    sign = (char)(Received[nrSign]);
7    value = (uint8_t)(Received[nrValue]);
8
9    if(axis == 'x')
10   {
11     xsign = sign;
12     xprevious = xvalue;
13     xvalue = value;
14   }
15
16   if(axis == 'y')
17   {
18     yprevious = yvalue;
19     yvalue = value;
20     if(abs(value - xprevious) < 2)
21       yvalue = yprevious;
22     else if(value > 100)
23       yvalue = yprevious;
24     else if(value < 50)
25       yvalue = yprevious;
26   }
27   HAL_UART_Receive_DMA(&huart2, &Received, BUFSIZE);
28 }
```

7 Zadania niezrealizowane

Nie zostało zrealizowane przekazanie napędu z silnika i serwomechanizmu na mechanizm napędowy oraz na ten służący do skręcania. W pierwszym przypadku jest to spowodowane brakiem czasu, wynikający ze zbyt długim poszukiwaniem rozwiązania na problem przeniesienia napędu z silnika do przekładni, natomiast w drugim tym, że wał serwomechanizmu ma stępione zębatki co uniemożliwia przekazanie jakiejkolwiek siły na dalszy podzespół.

8 Podsumowanie

Udało się zrealizować większość zadań. Nastąpiły drobne zmiany koncepcyjne jak użycie potencjometru do regulacji prędkości obrotowej napędu. To będzie wymagać mniejszej ingerencji gdy będziemy projektować regulator PID.

Literatura